ADT PLM

Programmer's Learning Machine

Matthieu Nicolas

IJD Seminar, 2016-02-02

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. To a web app
   - Goals
   - PLM as a library
   - Outcome
3. Assessment of user's code
   - Challenges
   - Extraction of the execution component
   - Docker
4. Result
5. Next steps

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. To a web app
   - Goals
   - PLM as a library
   - Outcome
3. Assessment of user's code
   - Challenges
   - Extraction of the execution component
   - Docker
4. Result
5. Next steps

Presentation of PLM

Purposes

- Application to learn programming.

Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...

Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble.

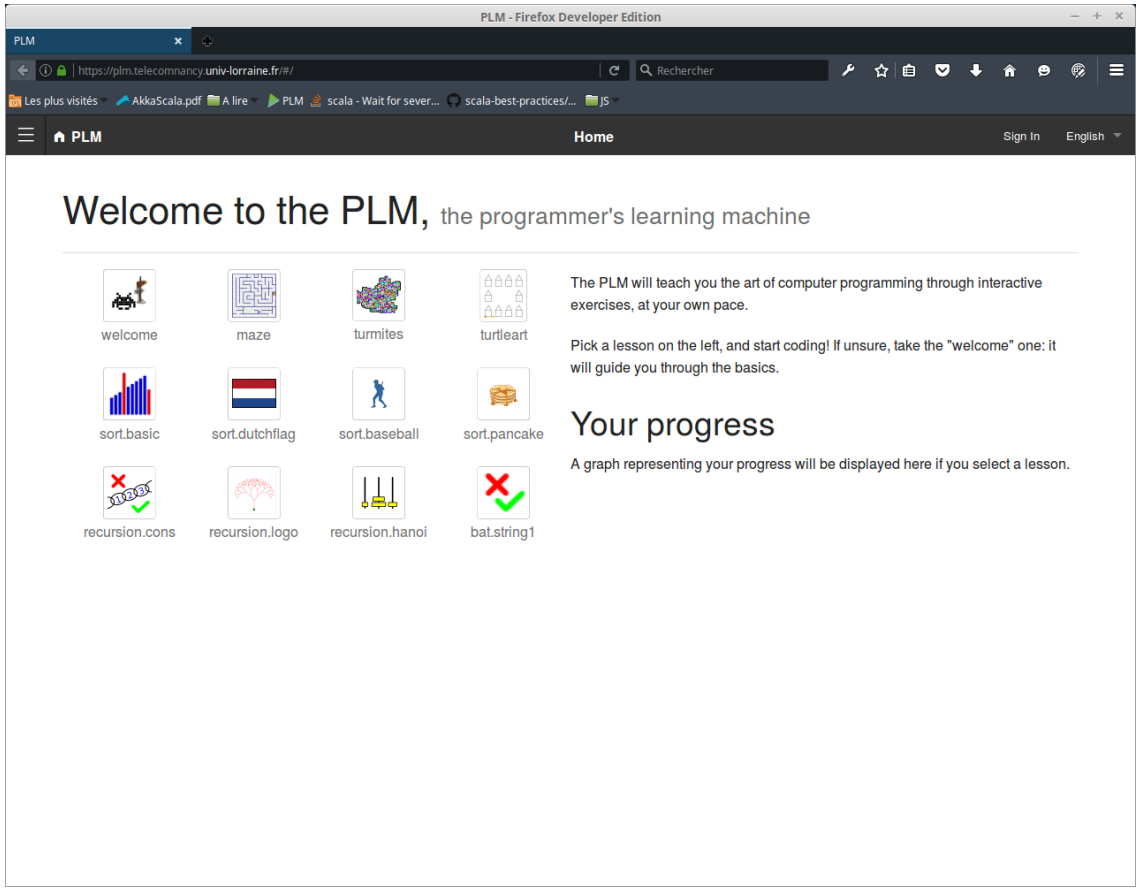
Presentation of PLM

Purposes

- Application to learn programming.
- Allows students to progress at their own speed...
- ... while the teacher helps the ones having trouble.
- Used at TELECOM Nancy since 2008.

Presentation of PLM

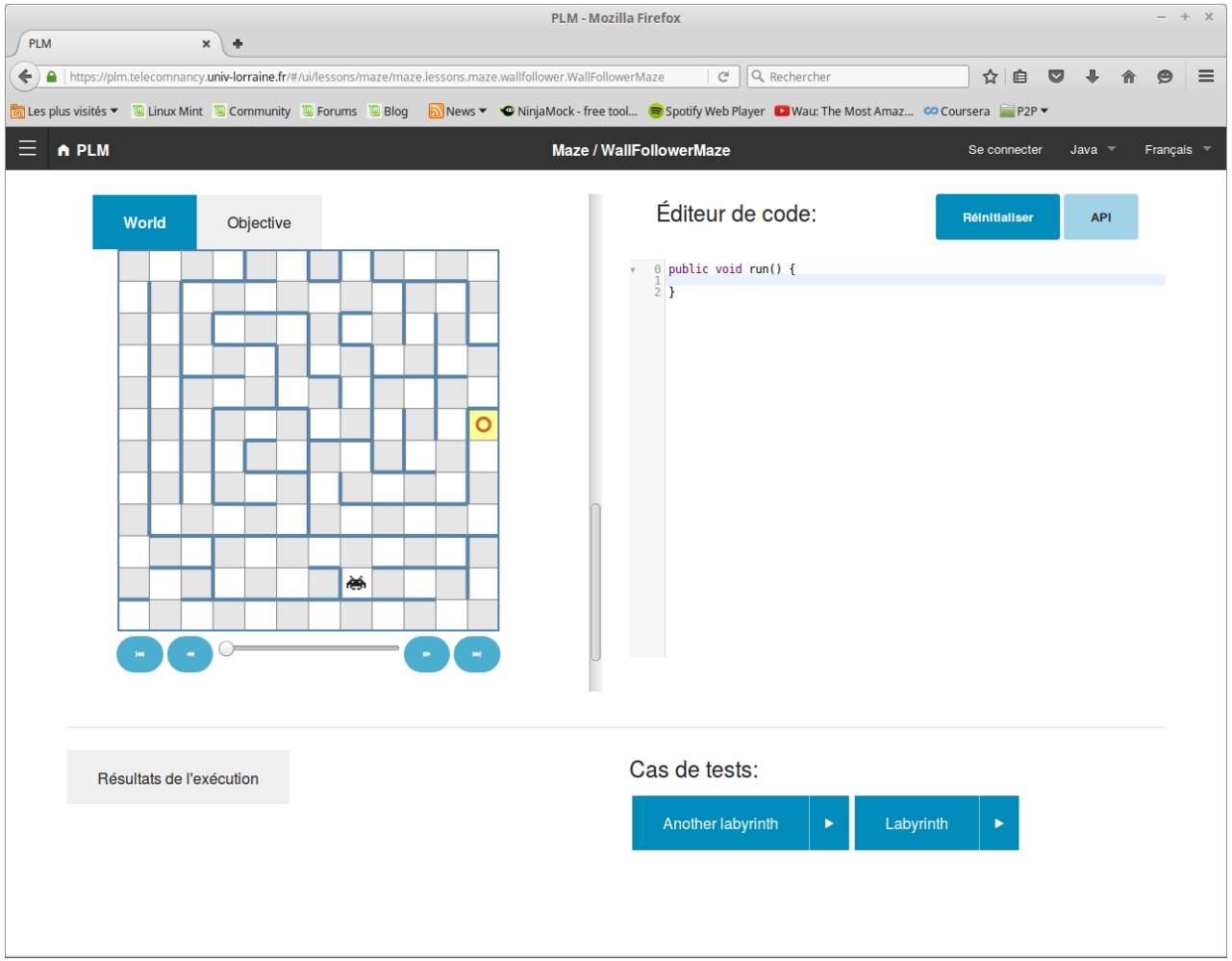
Quick demo



- Available at `https://plm.telecomnancy.univ-lorraine.fr`

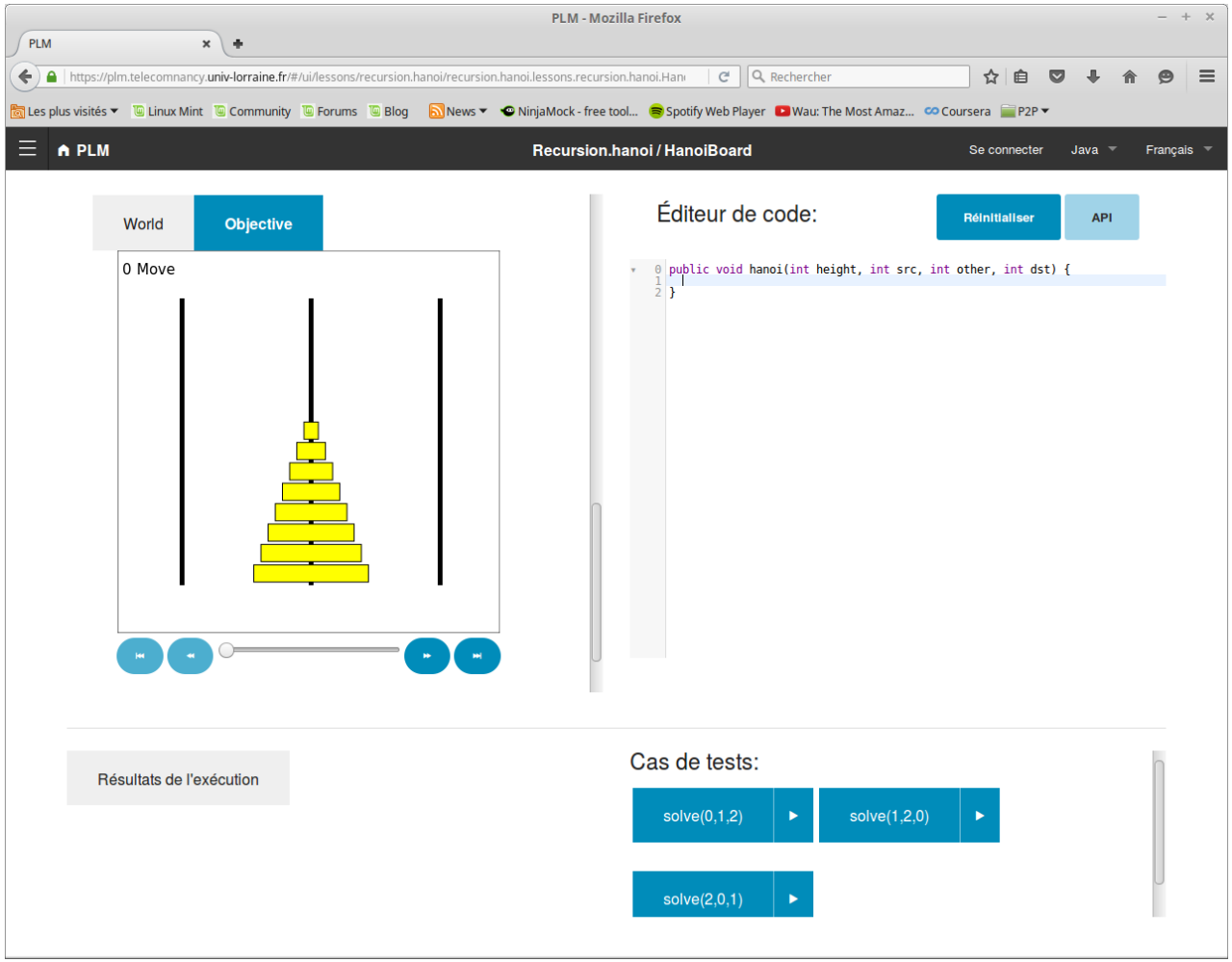
Presentation of PLM

12 lessons, 200 exercises



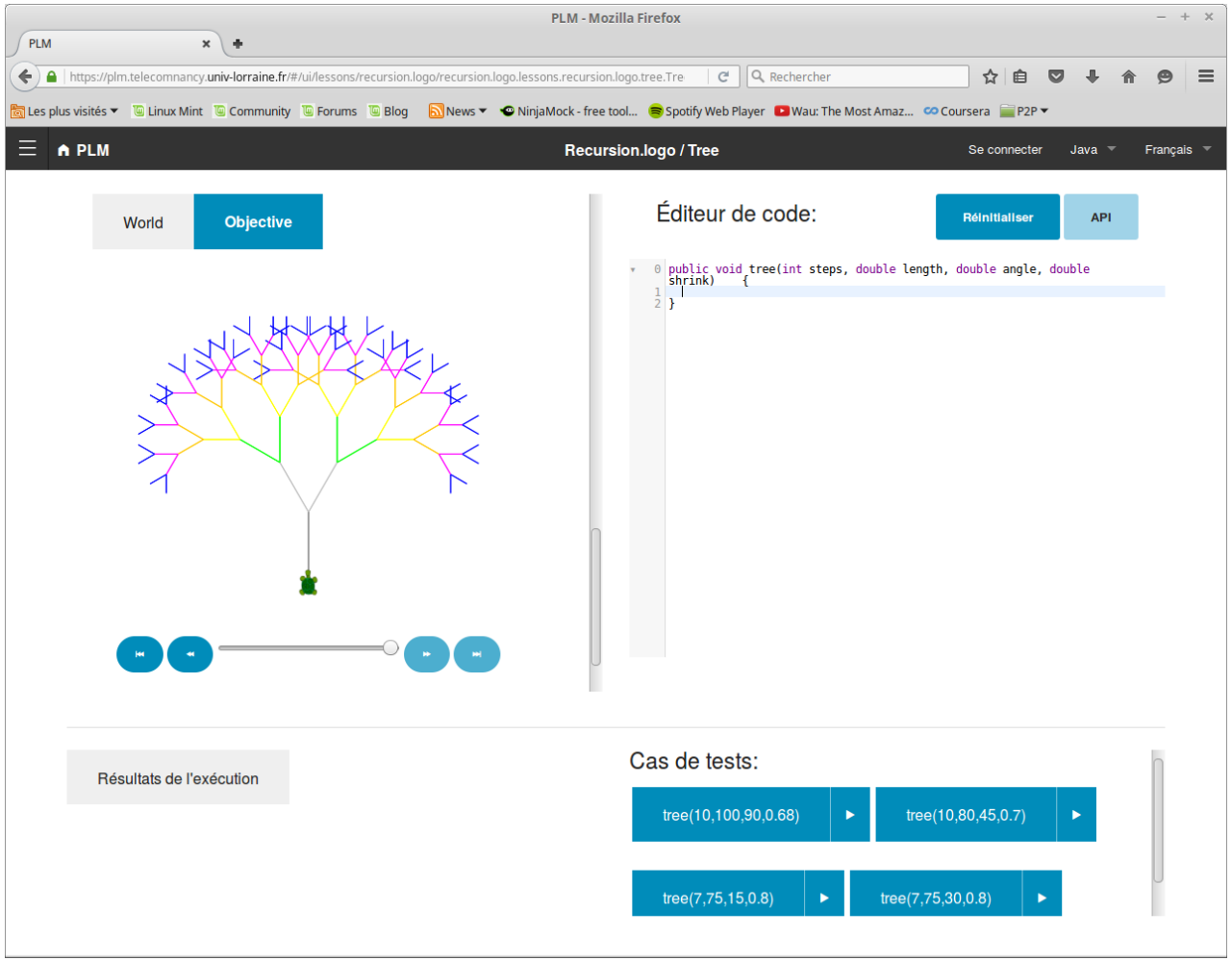
Presentation of PLM

12 lessons, 200 exercises



Presentation of PLM

12 lessons, 200 exercises

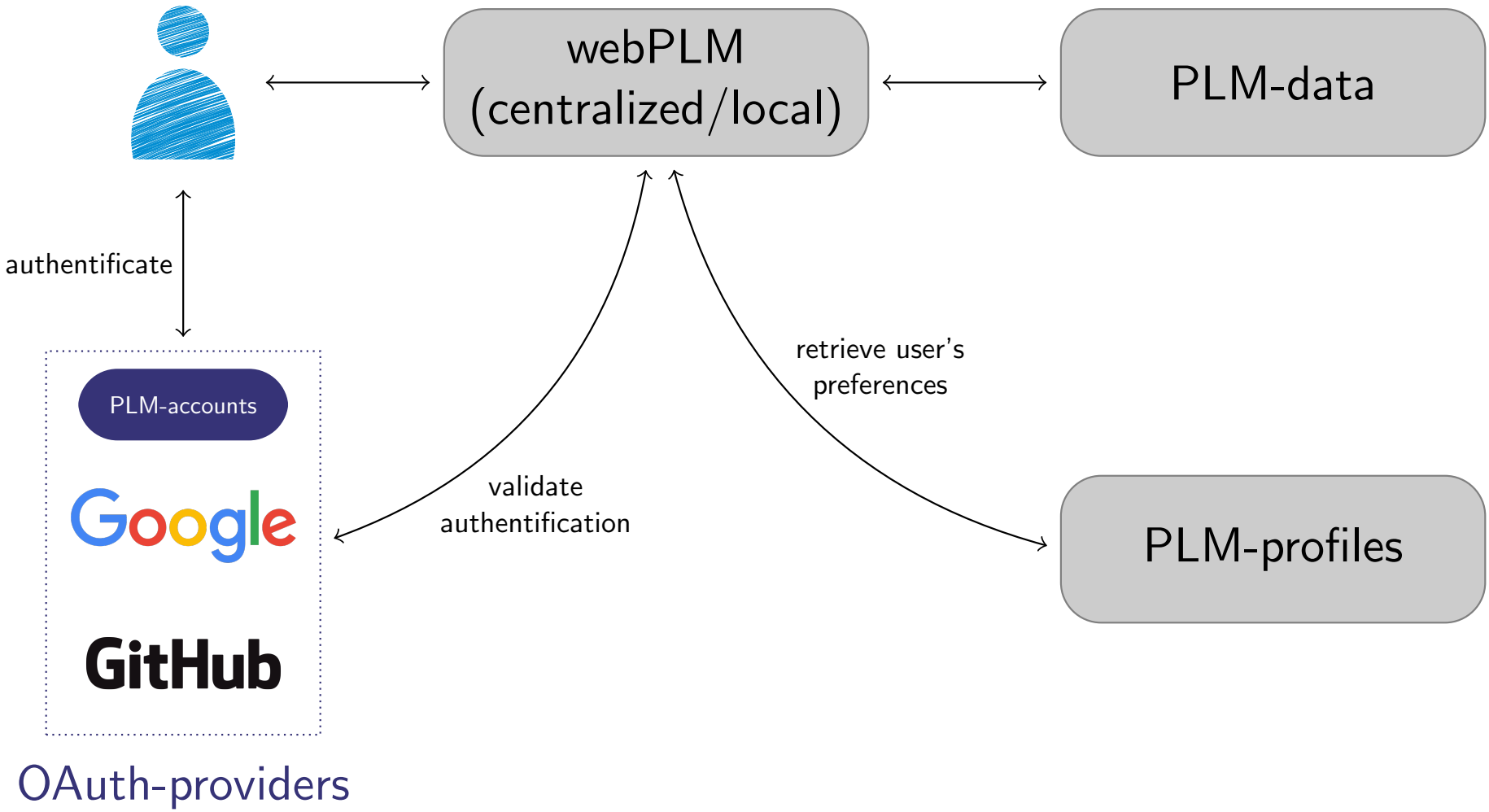


- Available languages:
  - English
  - French
  - Brazilian Portuguese

- Supported programming languages:

Presentation of PLM

Application's architecture



Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. To a web app
   - Goals
   - PLM as a library
   - Outcome
3. Assessment of user's code
   - Challenges
   - Extraction of the execution component
   - Docker
4. Result
5. Next steps

To a web app

Evolution of the project

- Formerly a fat client
  - Written in Java

To a web app

Evolution of the project

- Formerly a fat client
  - Written in Java
- Switch to a web application
  - Server implemented in Scala using *PlayFramework*
  - User interface written in Javascript using *AngularJS* and *Foundation*

To a web app

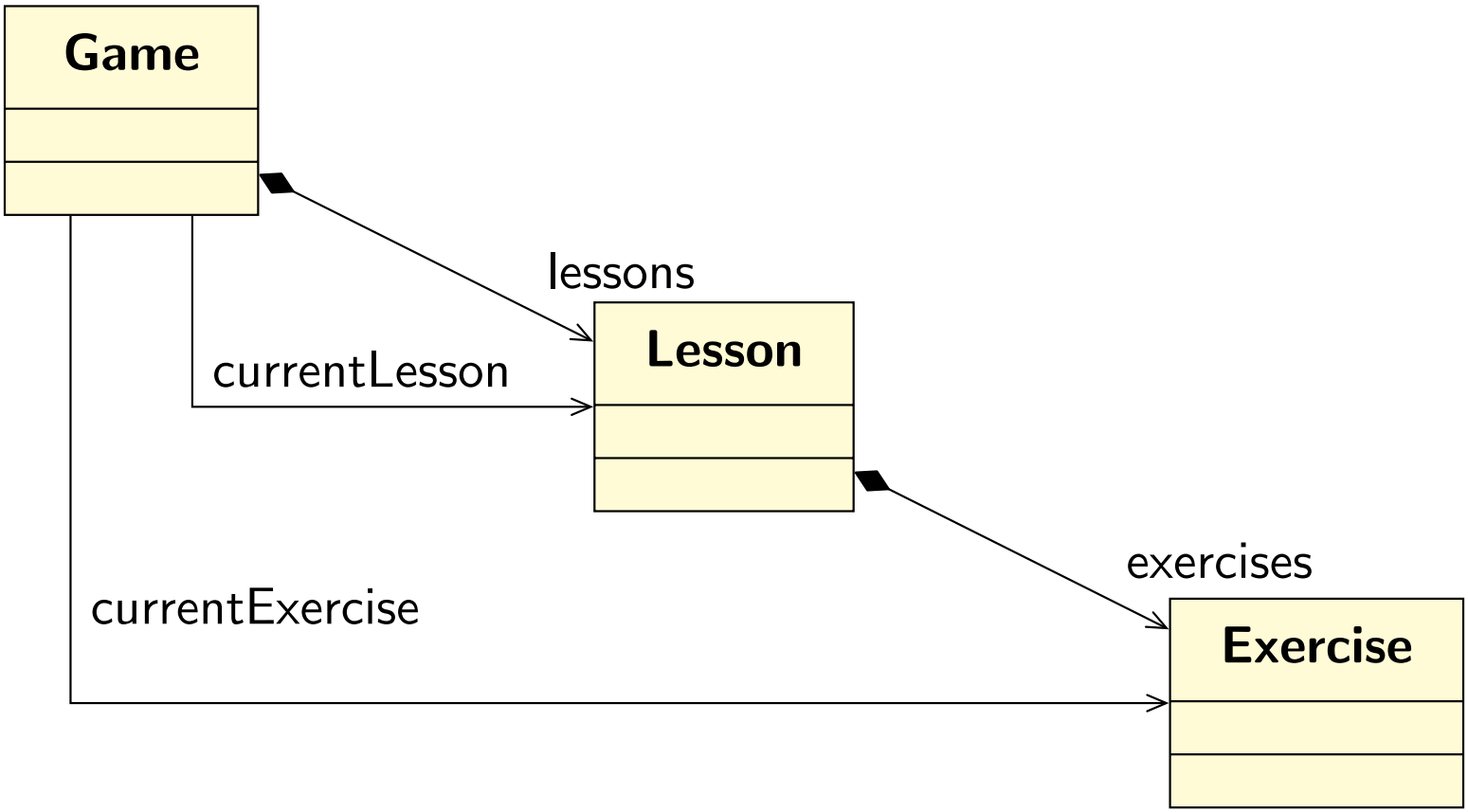
Motivations

- Want to switch to SaaS[1]
  - Easy to use
  - Easy to update
  - Easy to track usage data
- More user-friendly
- Aim to setup SPOC[2] and MOOC[3]

- But don't have the time and means for a reboot

[1] Software as a Service
[2] Small Private Online Course
[3] Massive Open Online Course

- Implemented a headless version of PLM: *PLM-engine*
  - Provide all PLM's content and methods
  - But without a user interface



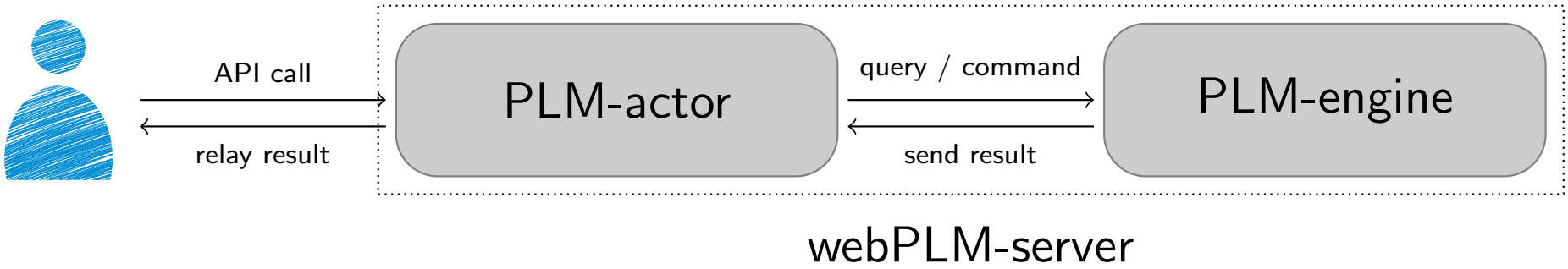
To a web app

Implementing the server

- Designed an API over PLM-engine
- Only need to implement a controller
  - Verify calls received from the client
  - Query or command PLM-engine according to the call
  - Send back result or acknowledgement to the client

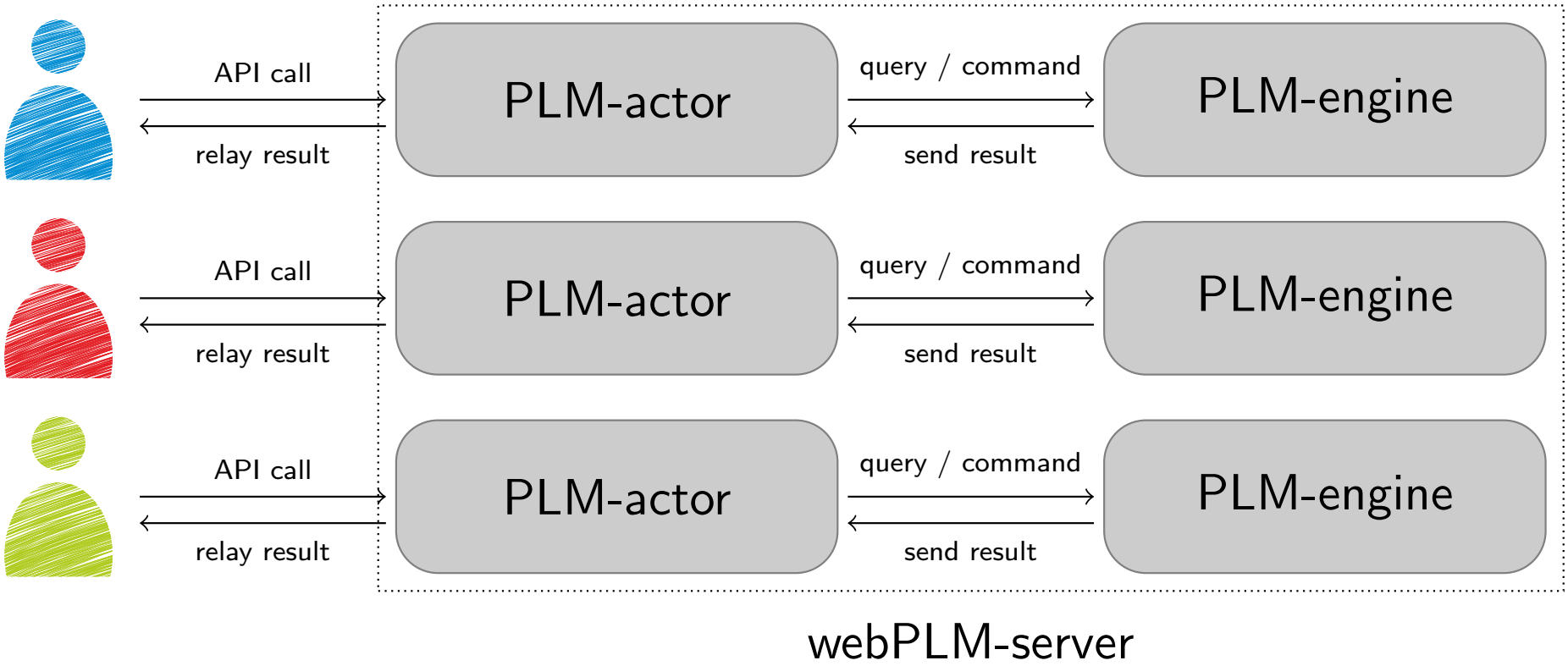
To a web app

Interactions between components



To a web app

Multi-user scenario

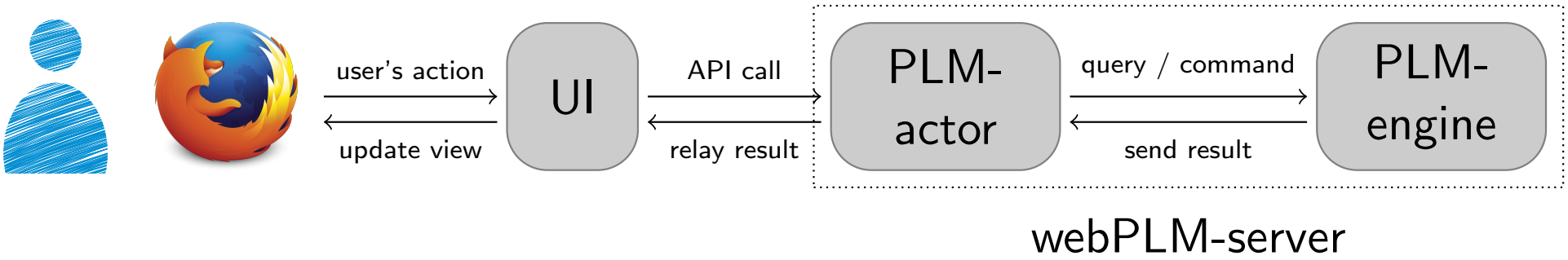


To a web app

Results

- Build quickly a web server from the fat client...
- ... but we also need a user interface

To a web app


user's action
update view
UI
API call
relay result
PLM-
actor
query / command
send result
PLM-
engine
webPLM-server

To a web app

- Have to translate user's actions into API calls
- Have to re-implement PLM-engine's data models

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. To a web app
   - Goals
   - PLM as a library
   - Outcome
3. Assessment of user's code
   - Challenges
   - Extraction of the execution component
   - Docker
4. Result
5. Next steps

Assessment of user's code

Limits

- Run on the same machine, same JVM

Assessment of user's code

Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops

Assessment of user's code

Limits

- Run on the same machine, same JVM

- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Endless file creation

Assessment of user's code

Limits

- Run on the same machine, same JVM

- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Endless file creation
- And from *System.exit(whatever)*?

Assessment of user's code

Limits

- Run on the same machine, same JVM
- How to protect ourselves from users' rookie mistakes?
  - Infinite loops
- And from more malicious "mistakes"?
  - Infinite thread creation
  - Endless file creation
- And from *System.exit(whatever)*?
- Scalability issues

Assessment of user's code

Delegate execution to workers

- Called *Judges* in the litterature
- Use PLM-engine as well

- Workflow:
  - Retrieve an execution request
  - Parse the request to extract parameters
  - Configure PLM-engine according to them
  - Run the user's code
  - Send back result to webPLM

Assessment of user's code

Message queues

- Message-driven architecture
- Loosely coupled system
- Asynchronous/Synchronous
- Help to implement:
  - Producer/Consumer pattern
  - Request/Response pattern
- Different reliability patterns of the message processing:
  - Only one worker
  - At least one worker
  - All workers
- Easy to scale

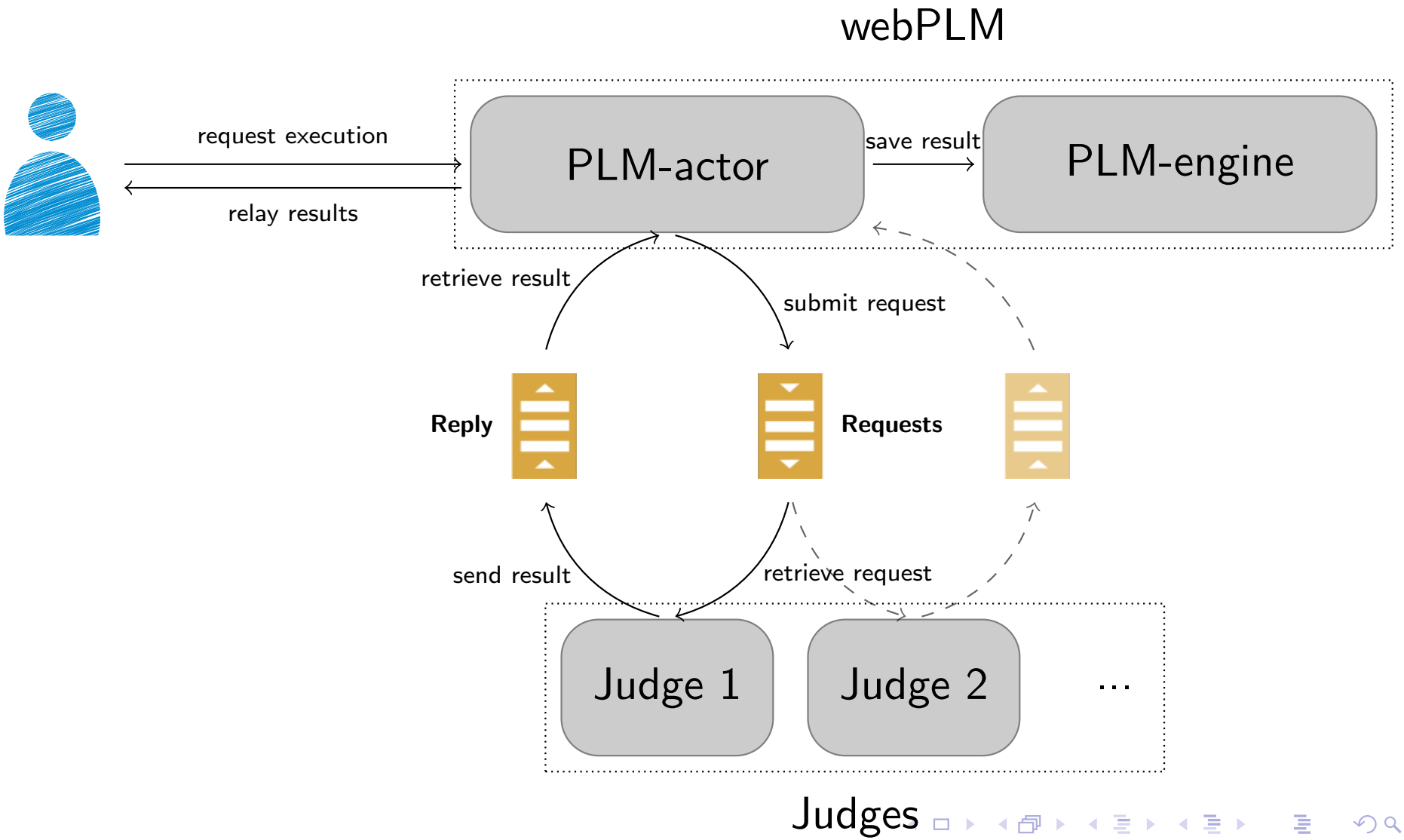
Assessment of user's code

Message queues

- Message-driven architecture
- Loosely coupled system
- Asynchronous/Synchronous
- Help to implement:
  - Producer/Consumer pattern
  - Request/Response pattern
- Different reliability patterns of the message processing:
  - Only one worker
  - At least one worker
  - All workers
- Easy to scale

Assessment of user's code

Architecture with judges



Assessment of user's code

Pros and cons

- Pros:
  - Allow to run code without impacting webPLM's performances
  - Meet the scalability requirements

Assessment of user's code

Pros and cons

- Pros:
  - Allow to run code without impacting webPLM's performances
  - Meet the scalability requirements
- Cons:
  - Make sure to use the right version of PLM-engine
  - Need to deploy them easily
  - Should restart them after each execution
  - Have to restrict their resources usage

Assessment of user's code

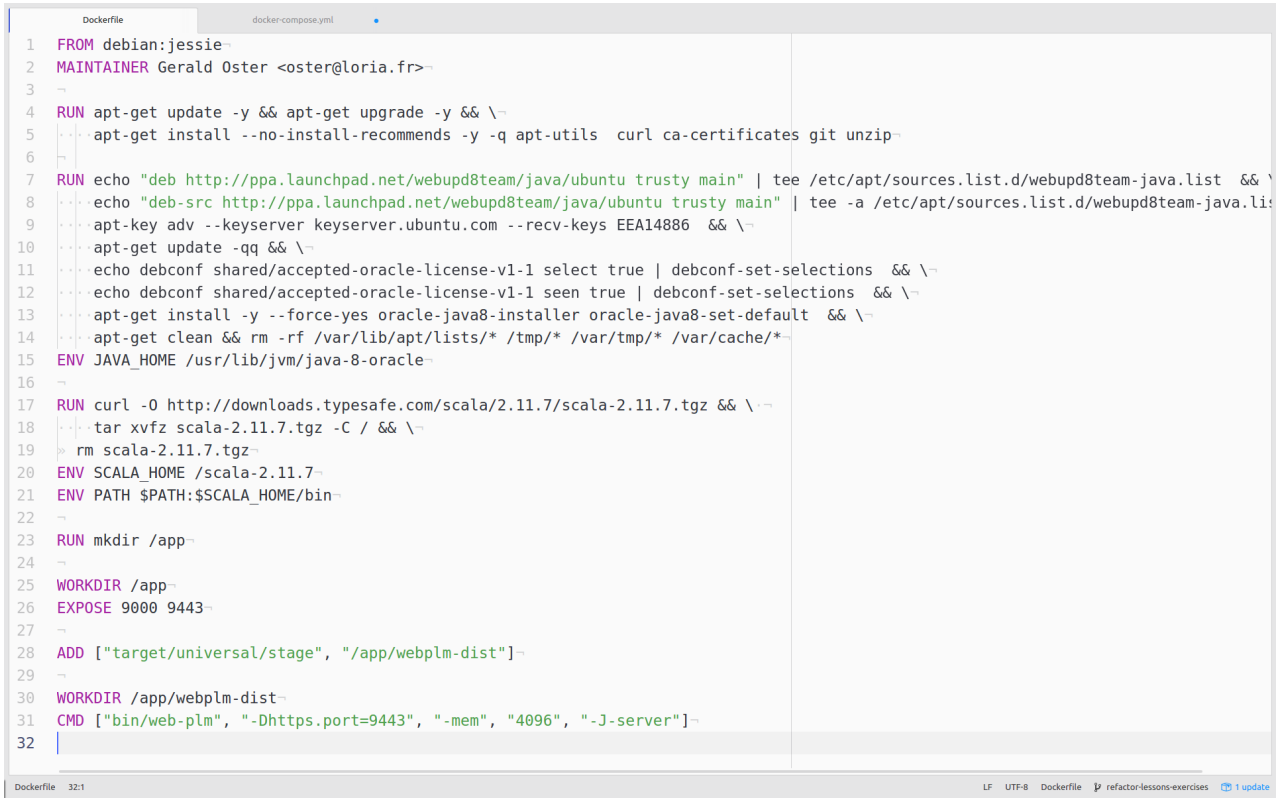
Docker

- Lightweight virtualization tool
- Build image of your application
- Run containers based on images

Assessment of user's code

Example of Dockerfile

- Dockerfiles describe how to set up the application



```
Dockerfile
1  FROM debian:jessie
2  MAINTAINER Gerald Oster <oster@loria.fr>
3
4  RUN apt-get update -y && apt-get upgrade -y && \
5      apt-get install --no-install-recommends -y -q apt-utils  curl ca-certificates git unzip
6
7  RUN echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee /etc/apt/sources.list.d/webupd8team-java.list  &&
8      echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee -a /etc/apt/sources.list.d/webupd8team-java.li
9      apt-key adv --keyserver keyserver.ubuntu.com --recv-keys EEA14886  && \
10     apt-get update -qq && \
11     echo debconf shared/accepted-oracle-license-v1-1 select true | debconf-set-selections  && \
12     echo debconf shared/accepted-oracle-license-v1-1 seen true | debconf-set-selections  && \
13     apt-get install -y --force-yes oracle-java8-installer oracle-java8-set-default  && \
14     apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/* /var/cache/*
15 ENV JAVA_HOME /usr/lib/jvm/java-8-oracle
16
17 RUN curl -O http://downloads.typesafe.com/scala/2.11.7/scala-2.11.7.tgz && \
18     tar xvfz scala-2.11.7.tgz -C / && \
19     rm scala-2.11.7.tgz
20 ENV SCALA_HOME /scala-2.11.7
21 ENV PATH $PATH:$SCALA_HOME/bin
22
23 RUN mkdir /app
24
25 WORKDIR /app
26 EXPOSE 9000 9443
27
28 ADD ["target/universal/stage", "/app/webplm-dist"]
29
30 WORKDIR /app/webplm-dist
31 CMD ["bin/web-plm", "-Dhttps.port=9443", "-mem", "4096", "-J-server"]
32
```

- Run *docker build -t tag /path/to/Dockerfile* to build the image
- Start containers with *docker run tag*

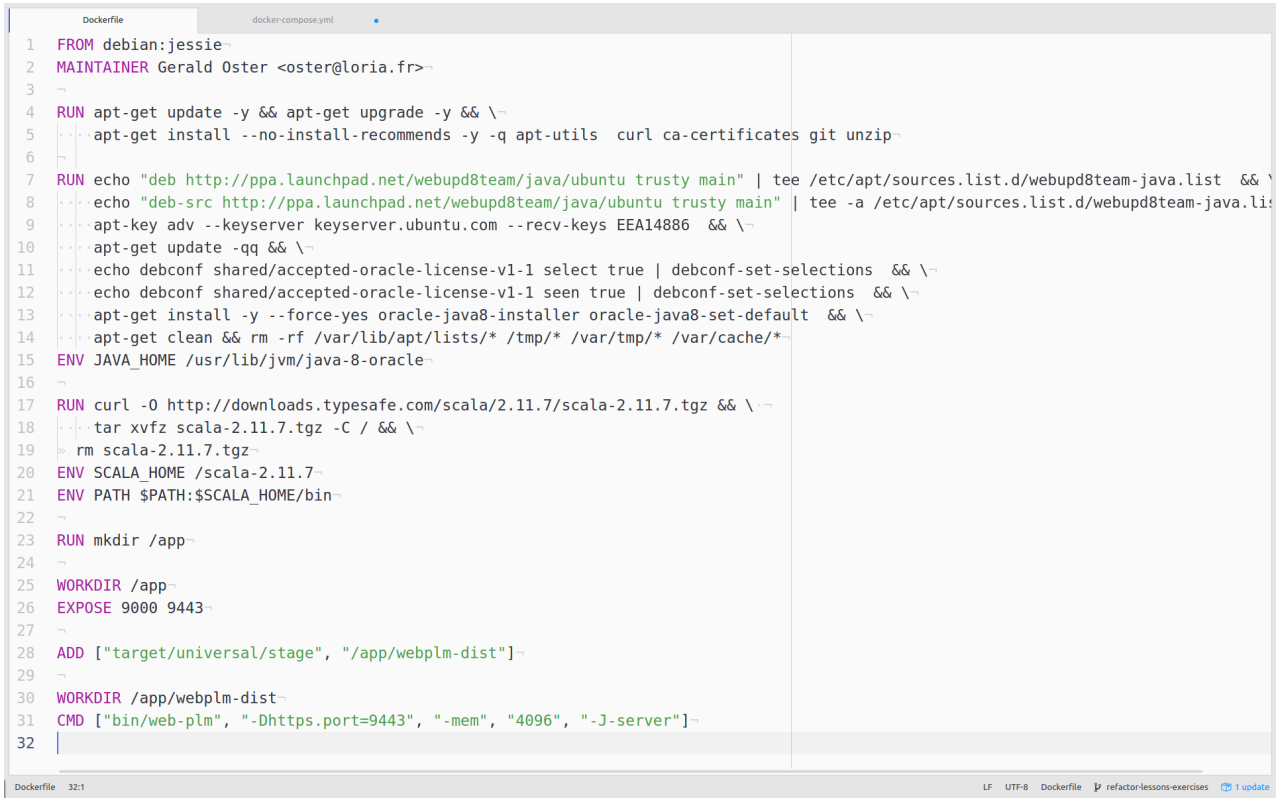
Dockerfile docker-compose.yml

```
FROM debian:jessie
MAINTAINER Gerald Oster <oster@loria.fr>

RUN apt-get update -y && apt-get upgrade -y && \
    apt-get install --no-install-recommends -y -q apt-utils  curl ca-certificates git unzip

RUN echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee /etc/apt/sources.list.d/webupd8team-java.list  && \
    echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee -a /etc/apt/sources.list.d/webupd8team-java.lis
    apt-key adv --keyserver keyserver.ubuntu.com --recv-keys EEA14886  && \
    apt-get update -qq && \
    echo debconf shared/accepted-oracle-license-v1-1 select true | debconf-set-selections  && \
    echo debconf shared/accepted-oracle-license-v1-1 seen true | debconf-set-selections  && \
    apt-get install -y --force-yes oracle-java8-installer oracle-java8-set-default  && \
    apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/* /var/cache/*
ENV JAVA_HOME /usr/lib/jvm/java-8-oracle

RUN curl -O http://downloads.typesafe.com/scala/2.11.7/scala-2.11.7.tgz && \
    tar xvfz scala-2.11.7.tgz -C / && \
  rm scala-2.11.7.tgz
ENV SCALA_HOME /scala-2.11.7
ENV PATH $PATH:$SCALA_HOME/bin

RUN mkdir /app

WORKDIR /app
EXPOSE 9000 9443

ADD ["target/universal/stage", "/app/webplm-dist"]

WORKDIR /app/webplm-dist
CMD ["bin/web-plm", "-Dhttps.port=9443", "-mem", "4096", "-J-server"]

```

Assessment of user's code

Example of Dockerfile

- Dockerfiles describe how to set up the application



```
Dockerfile    docker-compose.yml
1  FROM debian:jessie
2  MAINTAINER Gerald Oster <oster@loria.fr>
3
4  RUN apt-get update -y && apt-get upgrade -y && \
5      apt-get install --no-install-recommends -y -q apt-utils  curl ca-certificates git unzip
6
7  RUN echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee /etc/apt/sources.list.d/webupd8team-java.list  && \
8      echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee -a /etc/apt/sources.list.d/webupd8team-java.li
9      apt-key adv --keyserver keyserver.ubuntu.com --recv-keys EEA14886  && \
10     apt-get update -qq && \
11     echo debconf shared/accepted-oracle-license-v1-1 select true | debconf-set-selections  && \
12     echo debconf shared/accepted-oracle-license-v1-1 seen true | debconf-set-selections  && \
13     apt-get install -y --force-yes oracle-java8-installer oracle-java8-set-default  && \
14     apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/* /var/cache/*
15 ENV JAVA_HOME /usr/lib/jvm/java-8-oracle
16
17 RUN curl -O http://downloads.typesafe.com/scala/2.11.7/scala-2.11.7.tgz && \
18     tar xvfz scala-2.11.7.tgz -C / && \
19     rm scala-2.11.7.tgz
20 ENV SCALA_HOME /scala-2.11.7
21 ENV PATH $PATH:$SCALA_HOME/bin
22
23 RUN mkdir /app
24
25 WORKDIR /app
26 EXPOSE 9000 9443
27
28 ADD ["target/universal/stage", "/app/webplm-dist"]
29
30 WORKDIR /app/webplm-dist
31 CMD ["bin/web-plm", "-Dhttps.port=9443", "-mem", "4096", "-J-server"]
32
Dockerfile  32:1   LF  UTF-8  Dockerfile  refactor-lessons-exercises  1 update
```

- Run *docker build -t tag /path/to/Dockerfile* to build the image
- Start containers with *docker run tag*

Assessment of user's code

More about *docker run*

- Can also manage
  - Ports

Assessment of user's code

More about *docker run*

- Can also manage
  - Ports
  - Volumes

Assessment of user's code

More about *docker run*

- Can also manage
  - Ports
  - Volumes
  - Links between containers

Assessment of user's code

More about *docker run*

- Can also manage
  - Ports
  - Volumes
  - Links between containers
  - Environment variables
  - Runtime constraints on resources
  - Restart policies
  - And a **lot more**

Assessment of user's code

More about *docker run*

- Can also manage
  - Ports
  - Volumes
  - Links between containers
  - Environment variables
  - Runtime constraints on resources
  - Restart policies
  - And a **lot more**

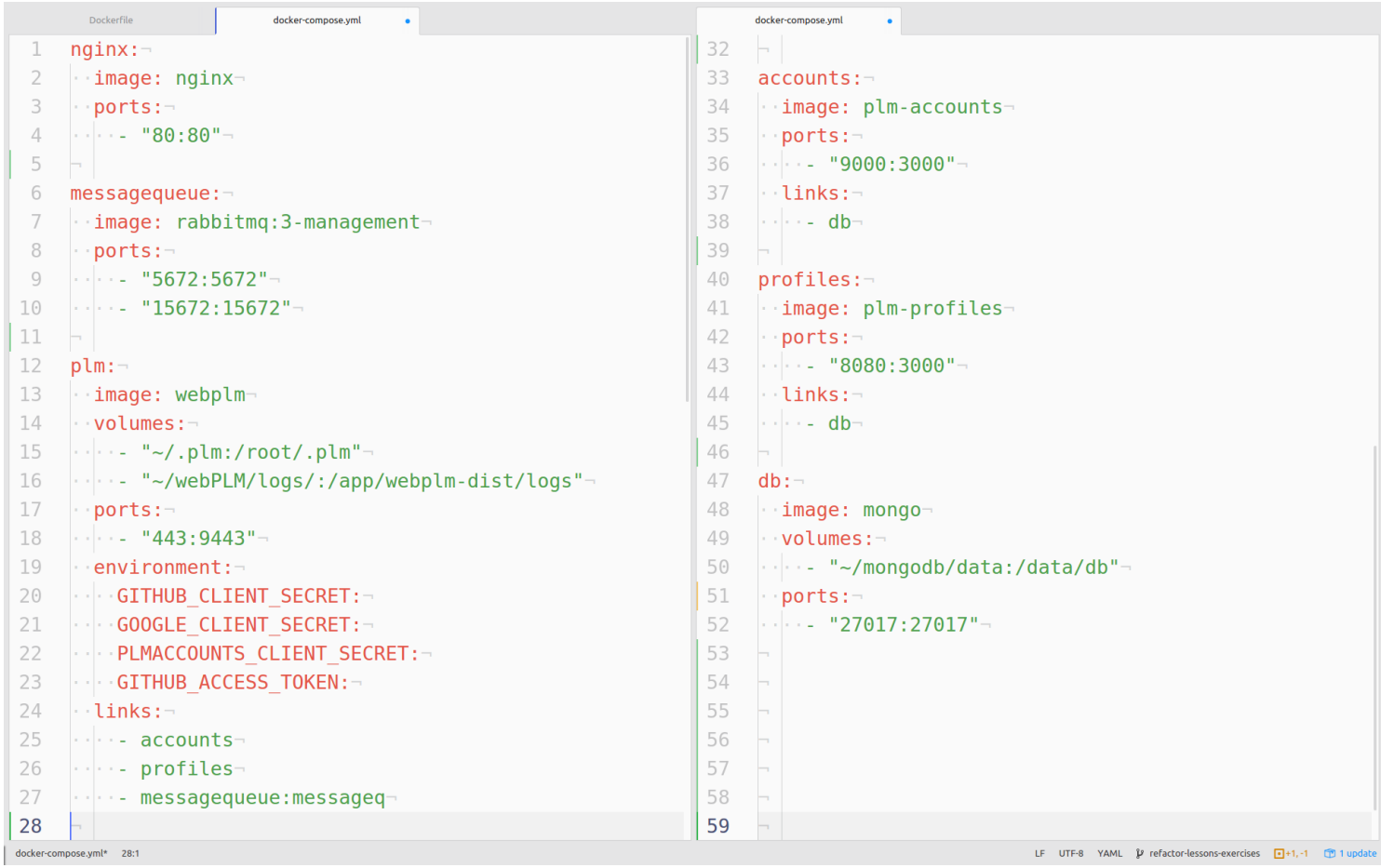
- Commands can become quite complex

docker run -p 443:9443 -link plm-accounts:accounts -v ~/webPLM/logs/:/app/webplm-dist/logs webPLM

Assessment of user's code

Docker-compose

- Tool to easily deploy multi-containers applications



Dockerfile | docker-compose.yml

```yaml
nginx:
  image: nginx
  ports:
    - "80:80"

messagequeue:
  image: rabbitmq:3-management
  ports:
    - "5672:5672"
    - "15672:15672"

plm:
  image: webplm
  volumes:
    - "~/.plm:/root/.plm"
    - "~/webPLM/logs/:/app/webplm-dist/logs"
  ports:
    - "443:9443"
  environment:
    GITHUB_CLIENT_SECRET:
    GOOGLE_CLIENT_SECRET:
    PLMACCOUNTS_CLIENT_SECRET:
    GITHUB_ACCESS_TOKEN:
  links:
    - accounts
    - profiles
    - messagequeue:messageq

accounts:
  image: plm-accounts
  ports:
    - "9000:3000"
  links:
    - db

profiles:
  image: plm-profiles
  ports:
    - "8080:3000"
  links:
    - db

db:
  image: mongo
  volumes:
    - "~/mongodb/data:/data/db"
  ports:
    - "27017:27017"
```

docker-compose.yml* 28:1 LF UTF-8 YAML refactor-lessons-exercises +1, -1 1 update

- Deploy environment with *docker-compose up*

Dockerfile | docker-compose.yml

```yaml
nginx:
  image: nginx
  ports:
    - "80:80"

messagequeue:
  image: rabbitmq:3-management
  ports:
    - "5672:5672"
    - "15672:15672"

plm:
  image: webplm
  volumes:
    - "~/.plm:/root/.plm"
    - "~/webPLM/logs/:/app/webplm-dist/logs"
  ports:
    - "443:9443"
  environment:
    GITHUB_CLIENT_SECRET:
    GOOGLE_CLIENT_SECRET:
    PLMACCOUNTS_CLIENT_SECRET:
    GITHUB_ACCESS_TOKEN:
  links:
    - accounts
    - profiles
    - messagequeue:messageq

```

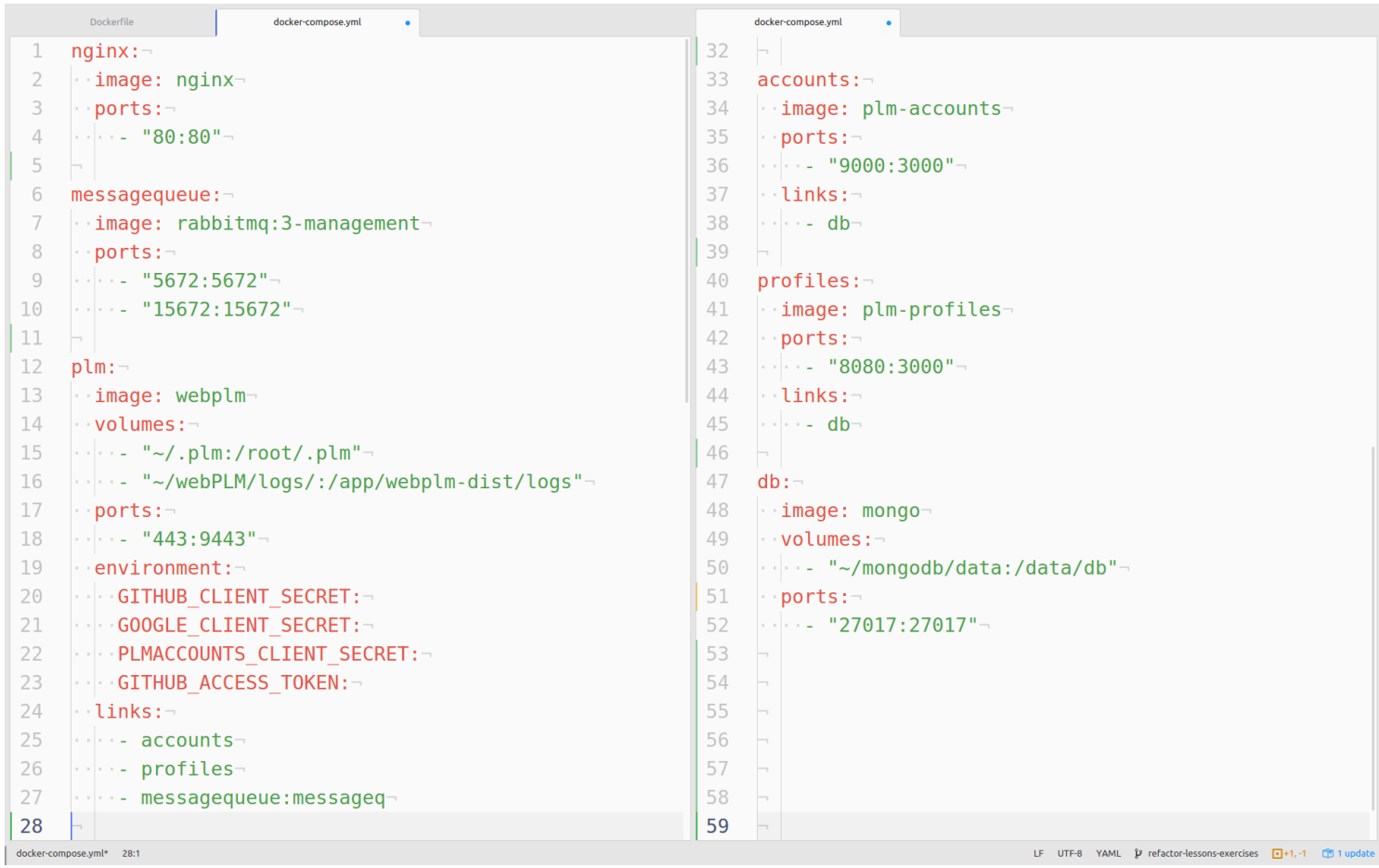
docker-compose.yml

```yaml

accounts:
  image: plm-accounts
  ports:
    - "9000:3000"
  links:
    - db

profiles:
  image: plm-profiles
  ports:
    - "8080:3000"
  links:
    - db

db:
  image: mongo
  volumes:
    - "~/mongodb/data:/data/db"
  ports:
    - "27017:27017"
```

Assessment of user's code

Docker-compose

- Tool to easily deploy multi-containers applications



Dockerfile | docker-compose.yml

```yaml
nginx:
  image: nginx
  ports:
    - "80:80"

messagequeue:
  image: rabbitmq:3-management
  ports:
    - "5672:5672"
    - "15672:15672"

plm:
  image: webplm
  volumes:
    - "~/.plm:/root/.plm"
    - "~/webPLM/logs/:/app/webplm-dist/logs"
  ports:
    - "443:9443"
  environment:
    GITHUB_CLIENT_SECRET:
    GOOGLE_CLIENT_SECRET:
    PLMACCOUNTS_CLIENT_SECRET:
    GITHUB_ACCESS_TOKEN:
  links:
    - accounts
    - profiles
    - messagequeue:messageq

accounts:
  image: plm-accounts
  ports:
    - "9000:3000"
  links:
    - db

profiles:
  image: plm-profiles
  ports:
    - "8080:3000"
  links:
    - db

db:
  image: mongo
  volumes:
    - "~/mongodb/data:/data/db"
  ports:
    - "27017:27017"
```

docker-compose.yml* 28:1 LF UTF-8 YAML refactor-lessons-exercises +1, -1 1 update

- Deploy environment with *docker-compose up*

Assessment of user's code

Docker in our case

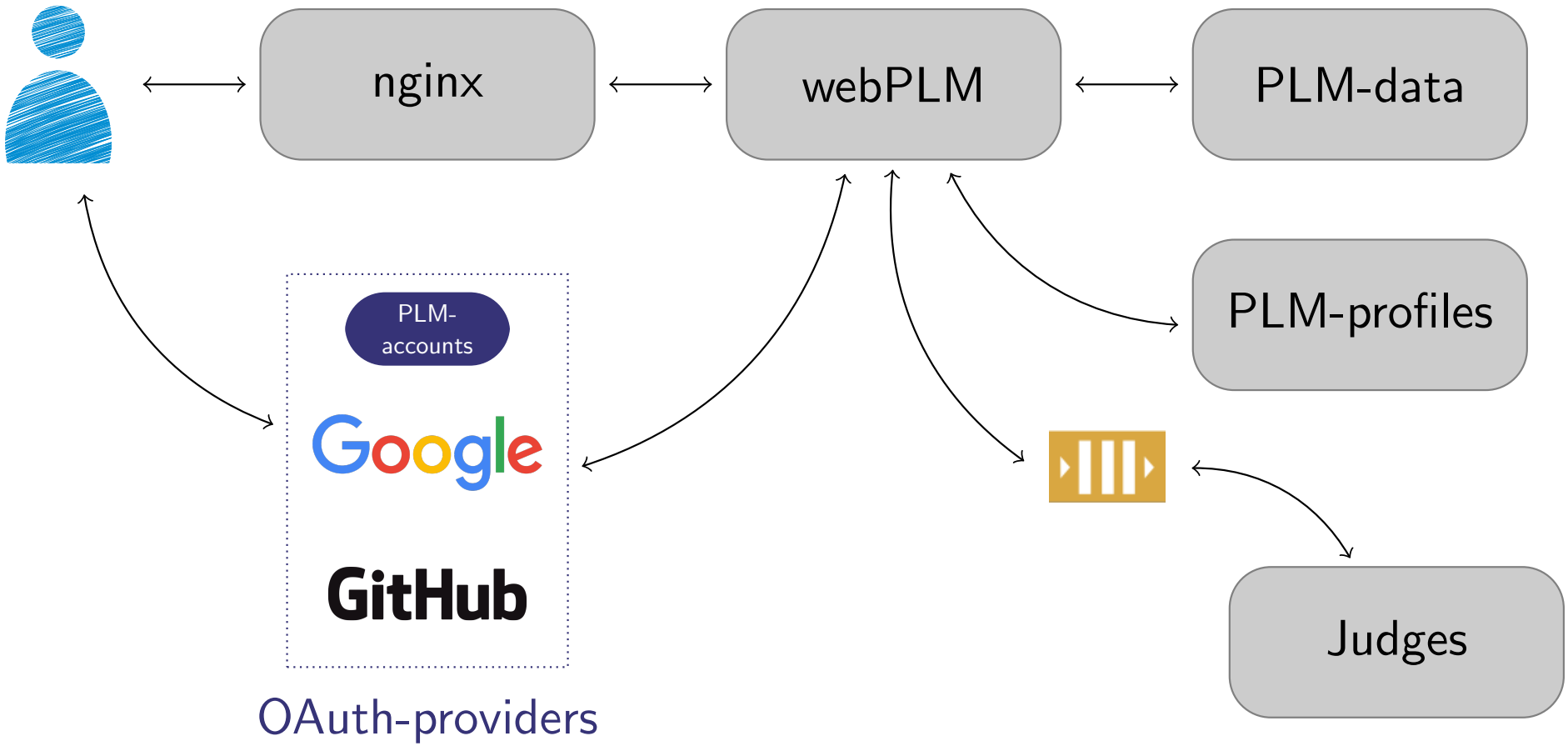
- Deploy easily all components
- Restart judges automatically
- Limit users' mischiefs

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. To a web app
   - Goals
   - PLM as a library
   - Outcome
3. Assessment of user's code
   - Challenges
   - Extraction of the execution component
   - Docker
4. Result
5. Next steps

Result

Current architecture



Result

Live-session in TELECOM Nancy

- Used in TELECOM Nancy in September 2015
- 30 hours of live testing with 100 students

Result

Live-session in TELECOM Nancy

- Used in TELECOM Nancy in September 2015
- 30 hours of live testing with 100 students

- Engine is (almost) working fine...
- ... but user experience needs to be improved!

Result

Live-session in TELECOM Nancy

- Scalability issues:
  - Work well with small exercises
  - Can't cope with workload of larger exercises

Result

Live-session in TELECOM Nancy

- Scalability issues:
  - Work well with small exercises
  - Can't cope with workload of larger exercises
- No tools for monitoring set up...

Result

Live-session in TELECOM Nancy

- Scalability issues:
  - Work well with small exercises
  - Can't cope with workload of larger exercises
- No tools for monitoring set up...
- ... so the bottleneck is unknown.

Outline

1. Presentation of PLM
   - Purposes
   - Demo
   - About PLM
   - Architecture
2. To a web app
   - Goals
   - PLM as a library
   - Outcome
3. Assessment of user's code
   - Challenges
   - Extraction of the execution component
   - Docker
4. Result
5. Next steps

Next steps

Refactor the code

- Rushed to release a stable version before September 2015...
- Needed to clean some parts of the code
- Merged local and centralized mode branches

Next steps

Simplify workflow to adapt the content

- Store most of content inside PLM-engine
- Heavy and error prone workflow
- Need to extract the content from PLM-engine's jar
- Allow to implement an exercise editor

Next steps

Solve performance issues

- Set up some monitoring tools
- Perform some load testing to identify the bottleneck

Next steps

- Integrate interns' contributions
- Set up Continuous Deployment
- Support additional programming languages
- Implement a debug mode similar to popular IDEs' ones
- Add features to help teachers to supervise their students
- ...

Questions

Thanks for your attention, any questions?